

## Malen statt programmieren

*Anwendungsintegration auf Basis von UML-Modellen folgt der Idee, dass ein Bild mehr sagt, als tausend Befehle in einer Programmiersprache*  
von Dr. Ralf Banning

Jede IT-Abteilung kennt das Problem: Ein Fachbereich wird umstrukturiert, Prozesse werden angepasst – die IT möge diese bitte entsprechend umsetzen, und zwar schnell. Während die Geschäftsanforderungen immer anspruchsvoller werden, fällt es Technikern oft schwer zu erklären, warum Prozesse, die auf Präsentationsfolien einfach aussehen, monatelange Entwicklungsarbeiten nach sich ziehen – das Dilemma der Anwendungsintegration. Die gilt weiterhin als riskant und heikel.

Seit der Erfindung der ersten Compiler durch Grace Hopper und John Backus wurden die Produktivität und Qualität der Programmierarbeit durch Automatisierung, Erhöhung des Abstraktionsgrades und verbesserte Wiederverwendbarkeit stetig gesteigert. Heute weisen Architekturkonzepte wie Serviceorientierung oder Event Processing über die Entwicklung einzelner Anwendungen hinaus. Der erhöhte Integrationsbedarf von serviceorientierten oder ereignisbasierten Architekturen (SOA, EDA) verlangt allerdings auch Ansätze, die mehr Transparenz und Flexibilität und schließlich Agilität auf Unternehmensebene bieten.

Der Leitsatz „Ein Bild sagt mehr als tausend Worte“ kennzeichnet die Welt der grafischen Modellsprachen als Alternative zu textuellen Programmiersprachen, beispielsweise der Unified Modeling Language (UML), die seit nunmehr zehn Jahren von der Object Management Group (OMG) propagiert wird. UML hat als Standard für den Entwurf von großen Software-Systemen weite Verbreitung erreicht. Mit der Model Driven Architecture (MDA) lancierte die OMG 2001 zusätzlich ein Framework, das diese Modellierungstechniken für die Software-Entwicklung besser und umfassender nutzbar machen soll

### **Modellierung der Schnittstellen**

Es liegt nahe, die Vorteile von MDA auch im Bereich der Anwendungsintegration zu nutzen. Man stößt hierbei allerdings auf das Problem, dass die Objekte klassischer UML und die der Anwendungsintegration auf verschiedenen Abstraktionsebenen liegen. Eine einfache Eins-zu-eins-Übertragung der Konzepte kann an der Heterogenität und Komplexität im Integrationsbereich rasch scheitern. Abhelfen können spezielle UML-Profile. 2004 wurde ein solches UML-Profil für Enterprise Application Integration (EAI) veröffentlicht.

Das in den letzten zwei Jahren eingeführte Konzept einer Model Driven Integration (MDI) beschreibt einen von MDA abgeleiteten Integrationsansatz, bei dem Schnittstellen komplett auf der Basis grafischer und direkt ausführbarer Modelle umgesetzt werden. Im Gegensatz zur Code-Generierung bei MDA werden UML-Modelle bei MDI direkt kompiliert und in einer UML Virtual Machine ausgeführt: Die Modelle sind gewissermaßen der Code. Schnittstellen sind so immer vollständig dokumentiert, die Wartung vereinfacht sich drastisch und die Abhängigkeit eines Unternehmens vom einzelnen Entwicklern kann deutlich reduziert werden.

Bei traditionellen EAI Ansätzen war bislang für die Orchestrierung der Integrationservices eine Mischung aus der Anwendung proprietärer Konfigurationswerkzeuge und händischer Programmierung notwendig. Für einige EAI Tools existieren zwar grafische Oberflächen, aber deren Entwicklungen sind von dem Support eines Herstellers abhängig und lassen sich nicht standardisiert auslesen oder in anderer Form verfügbar machen. Proprietäre EAI-Notationen verursachen zudem eine hohe Abhängigkeit von einzelnen Mitarbeitern mit Spezialwissen. Bei MDI sind diese Probleme durch die Verwendung von UML erheblich reduziert.

Ein ebenso wichtiger Unterschied von EAI zu MDI ist der Adapter Approach. Wenn die Anforderungen über die Konfigurationsoptionen der fertigen Komponenten der EAI-Hersteller hinausgehen, müssen die Entwickler dafür individuelle Lösungen mit Software Development Kits erarbeiten.

Die Folge: Es treten all die bekannten Probleme der Softwareentwicklung auf, namentlich fehlende Dokumentation, aufwändige Wartung. Das Besondere an MDI ist nun, dass die Kalibrierung und Anpassung von Schnittstellen Teil des MDI-Konzepts sind: Sie werden ebenfalls über UML dargestellt und sind dadurch über ihre gesamte Lebensdauer viel leichter zu handhaben.

### **Ausführung von UML-Diagrammen**

Zu den wichtigsten Vorteilen von MDI für das Anwenderunternehmen zählen daher weiche Migrationspfade, langfristige Investitionssicherheit und kürzere Projektlaufzeiten. Kleine Änderungsanfragen lassen sich im Modell innerhalb von Minuten durchführen. Selbst komplexe Projekte dauern kaum länger als ein paar Wochen. Außerdem: Bei MDI lässt sich mit UML, im Gegensatz zu herkömmlichen Integrationsansätzen, auch Verarbeitungslogik abbilden. Dies vereinfacht die Absprache zwischen dem Business- und dem IT-Verantwortlichen. Die E2E Bridge des Schweizer Herstellers E2E ist die

erste hundertprozentig modellbasierte Integrationssoftware: ein Enterprise Service Bus (ESB) in Form einer Virtual Machine für UML. Dieser UML-ESB nutzt die fünf wichtigsten UML-Diagrammtypen: Use-Case-Diagramme dienen der Beschreibung von User-Rollen, Security-Modellen, Systemgrenzen und Services. Class-, Activity- und Deployment-Diagramme werden zum Design der eigentlichen Services genutzt, und während des Produktivbetriebs direkt ausgeführt. Sequence-Diagramme schließlich werden von der E2E Bridge automatisch beim Debugging generiert, um dem Entwickler einen exakten modellbasierten Debug Trace anzuzeigen.

### **MDI setzt einiges voraus**

Durch die UML-Modellierung sind zur Integration wesentlich weniger Ressourcen nötig, als bei herkömmlichen Integrationsansätzen. Denn Change Requests können nun zügig und wartungsfreundlich umgesetzt werden; das Anhängen neuer Funktion als Rucksack an bestehende Anwendungen gehört der Vergangenheit an. Anwendungsintegration mittels UML-Modellen ermöglicht eine ganzheitliche Betrachtung des Lebenszyklus, von der Aufnahme der Anforderungen, über das Design der einzelnen Komponenten, bis hin zum Produktionseinsatz in der operativen IT und dem nachfolgenden Change Management – sowohl für technische Detailverbesserungen als auch für umwälzende Prozessanpassungen.

Für die Einführung des MDI-Ansatzes in einem Unternehmen müssen freilich wichtige Voraussetzungen gegeben sein: Passen die UML im Allgemeinen und die ausgewählten Produkte im Besonderen in die Software-Strategie des Unternehmens? Können Mittel und Wissen bereitgestellt werden, um eine SOA-Architektur aufzubauen? Eine gewissenhafte Bestandsaufnahme der vorhandenen Lösungen und Anforderungen und eine klare Integrationsstrategie sind also auch für einen MDI Einsatz wesentliche Vorbedingungen.

### **Services bei MDI**

Beim Einsatz von MDI lässt sich die Entwicklungsarbeit in folgende Einzelschritte aufteilen:

1. Import bestehender Backend-Metadaten und –Funktionen
2. Design der Integrationsservices und deren Orchestrierung
3. Validierung und Kompilierung der Services in lauffähige Modelle
4. Installation der lauffähigen Services in die bestehende Server-Landschaft
5. Qualitätssicherung und Debugging

**Dr. Ralf Banning** ist Berater bei der conto-xo GmbH in Reutlingen und verantwortet dort den Bereich Integrationsarchitekturen.